Creative Technologies in the Classroom: Librería Arduino y firmata: Comunicación serial Arduino - Processing

Colegio Seminario Diocesano de Duitama - M&T

1. Librería Arduino para Processing

La librería `Arduino` para processing permite controlar una placa Arduino desde processing sin tener que escribir código de Arduino, simplemete con cargar un código pre - diseñado en la placa, se podrán controlar los pines de Arduino y obtener los resultados en processing.

La librería está disponible para su descarga en processing2-arduino.zip y su instalación es tan simple como descomprimir el contenido del `.zip` en la carpeta `libraries` del directorio `Arduino`.

NOTA: *En GNU/Linux será necesario cambiar `Arduino.jar` por `arduino.jar` debido a que GNU/Linux distingue entre mayúsculas y minúsculas*. `Arduino.jar` se encuentra dentro de la carpeta "library" de la librería `Firmata`.

Información detallada sobre la librería `Firmata` puede encontrarse en Arduino and Processing.

1.1. Preparando Arduino:

Para preparar la placa Arduino para comunicarse con Processing usando la librería firmata simplemente es necesario cargar el sketch `StandartFirmata` disponible en Arduino IDE en la ruta `Archivo ->Ejemplos ->Ejemplos desde Librerías ->Firmata ->StandartFirmata`. En esta misma ruta pueden encontrarse otros ejemplos de uso de firmata más específicos y elaborados, sin embargo `StandartFirmata` será suficiente para leer y escribir en todos los pines diginales y análogos de Arduino UNO.

1.2. Sketch Líneas Trigonométricas

En este sketch se hace una exploración genométrica de las funciones trigonométricas construyendo los respectivos segmentos en el círculo goniométrico, para tal fin se deben tener presente algunos puntos esenciales:

1. Como ya debe conocerse, el sistema coordenado de processing no es igual al sistema coordenado convencional, es decir, en un canvas de processing el punto (0, 0) se encuentra ubicado en la esquina superior izquierda y las coordenadas verticales (ordenadas) aumenta hacia abajo y las coordenadas horizontales (abscisas) como es acostumbrado, hacia la derecha.

2. La unidad de medida standar en processing es el pixel por lo que una unidad corresponde a un pixel.

3. Para estudiar y entender el código de este sketch debe conocerse lo fundamental sobre la definición de funciones y clases en processing.

4. **Construcción de los segmentos trigonométricos**

   a) **Segmento del Seno:** Segmento vertical que va desde el punto $(\cos(A), 0)$ hasta el punto $(\cos(A), \sin(A))$.
   b) **Segmento del Coseno:** Segmento horizontal entre los puntos $(0, 0)$ y $(\cos(A), 0)$.
   c) **Segmento de la Tangente:** Segmento vertical con extremos en los puntos $(1, 0)$ y $(1, \tan(A))$.
   d) **Segmento de la Cotangente:** Segemento horizontal entre $(0, 1)$ y $(\cot(A), 1)$.
   e) **Segmento de la Secante:** Segmento transversal desde $(0, 0)$ hasta $(1, \tan(A))$.
   f) **Segemento de la Cosecante:** Segmento transversal desde el origen hasta $(\cot(A), 1)$.

   Estos segmentos están definidos sobre el sistema coordenado convencional no sobre el sistema coordenado de processing.

Para comprender el código del Listing 1 se leerá de manera modular. En primer lugar las partes referentes a la comunicación Processing Arduino. En las líneas 1 a 3 se llama a las librerías de comunicación serial y firmata y se define el objeto arduino, luego en las líneas 38 a 40 se define el puerto y la velocidad de comunicación y se configuran los pines digitales 2 al 7 de arduino como pines de entrada, en estos se conecta el dip switch, a continuación en la línea 46 (en el `draw()`) se llama a la función `communication()` definida en las líneas 59 a 69, en ésta se leen los pines digitales del 2 al 8 y su estado se almacena en el vector `buttonState[]` y el valor del pin análogo A0 se almacena en la variable `angle`.

Luego, en las líneas 6 a 25 se definen todas la variables y objetos con las que se que construyen los segmentos trigonométricos, en primer lugar en la línea 6 se define un vector `boolean` para el estado de los botones del dip switch, las líneas 8 y 9 corresponde a un par de vectores de imágenes destinados a contener las imágenes correspondientes a los botones en estado activado y desactivado respectivamente, el arreglo `imFile[]` entre las líneas 10 a 15 contiene los nombres de los archivos de imágen de los botones previamente cargados en la carpeta `data` del sketch, el arreglo `buttonName[]` contiene los nombres de las funciones trigonométricas que se pondrán como textos nombre de los botones y el arreglo `colorFunction` define los colores correspondientes a cada segmento trigonométrico. La línea 24 define las coordenadas del centro del círculo goniométrico y la línea 25 crea un vector de objetos de la clase boton que se puede leer en el Listing 2.

La definición de los botones se desarrolla con el bucle **for** de las líneas 32 a 36 (en el `setup()`), allí se cargan las imágenes que representan los dos estados de cada botón y se define cada objeto del vector botones.

A continuación en el `draw()`, en la línea 47 se llama a la función `ejes()` que define el sistema de coordenadas para la construcción, se ubican los botones y se llama finalmente a la función `segments()` que dibuja cada segmento trigonométrico de acuerdo al sistema de coordenadas definido con la función ejes y respecto del sistema de coordenadas de processing.

La última función presente en el Listing 1, la función `monitoring()`, si se llama desde el `draw()` escribirá en la consola de processing el estado de cada botón y el valor del ángulo.

Cada vez que se active un switch en el dip switch se activará uno de los segmentos trigonométricos; el ángulo se controla con el potenciometro.

```
1  import processing.serial.*;
2  import cc.arduino.*;
3  Arduino arduino;
4  
5  // Botones
6  boolean[] buttonState = {false, false, false, false, false, false};
7  float angle = 0;
8  PImage buttonsOn[] = new PImage[6];
9  PImage buttonsOff[] = new PImage[6];
10 String imFile[] = {
11   "sinOn.png", "cosOn.png", "tanOn.png",
12   "cotOn.png", "secOn.png", "cscOn.png",
13   "sinOff.png", "cosOff.png", "tanOff.png",
14   "cotOff.png", "secOff.png", "cscOff.png"
15 };
16 String buttonName[] = {
17   "sin(A)", "cos(A)", "tan(A)",
18   "cot(A)", "sec(A)", "csc(A)"
19 };
20 color colorFunction[] = {
21   #BA6259, #55626E, #8F6CA0,
22   #598DAF, #59AB82, #CB9B4D
23 };
24 int h, k;
25 boton botones[] = new boton[6];
26 
27 void setup(){
28   size(500, 600);
29   textAlign(CENTER);
30   textSize(18);
31 
32   for (int i = 0; i < 6; i++){
33     buttonsOn[i] = loadImage(imFile[i]);
34     buttonsOff[i] = loadImage(imFile[i + 6]);
35     botones[i] = new boton(31 + i * 75, 524, buttonsOn[i], buttonsOff[i]);
36   }
37 
38   arduino = new Arduino(this, "/dev/ttyACM0", 57600);
39   for (int i = 2; i < 8; i++){
40     arduino.pinMode(i, Arduino.INPUT);
41   }
```

```
}

void draw(){
  background(#C4E9ED);
  communication();
  ejes(-2, 2, 0.25, -2, 2, 0.25);
  noStroke();
  fill(#0c657e);
  quad(0, height - 100, width, height - 100, width, height, 0, height);
  for (int i = 0; i < 6; i++){
    botones[i].dibuja(buttonState[i]);
    fill(colorFunction[i]);
    text(buttonName[i], 60 + i * 75, 580);
  }
  segments(buttonState);
}

void communication(){
  for (int i = 2; i < 8; i++){
    if (arduino.digitalRead(i) == Arduino.HIGH){
      buttonState[i - 2] = true;
    }
    else{
      buttonState[i - 2] = false;
    }
  }
  angle = map(arduino.analogRead(0), 0, 1023, 0, 2 * PI);
}

void ejes(int xMin, int xMax, float xStep, int yMin, int yMax, float yStep){
  int numberXDivisions = int((xMax - xMin) / xStep);
  int numberYDivisions = int((yMax - yMin) / yStep);
  h = width / 2;
  k =  height / 2 - 50;
  stroke(0);
  strokeWeight(1);
  line(0, k, width, k);
  for (int i = 0; i < numberXDivisions; i++){
    line(0 + i * width / numberXDivisions, k, 0 + i * width / numberXDivisions,
     k + 10);
  }
  line(width / 2, 0, width / 2, height - 100);
  for (int i = 0; i < numberYDivisions; i++){
    line(h - 10, 0 + i * (height - 100) / numberYDivisions, h, 0 + i * (height
   - 100) / numberYDivisions);
  }
  strokeWeight(2);
  noFill();
  ellipse(h, k, width / 2, width / 2);
}

```

```
void segments(boolean states[]){
  int r = width / 4;
  if (states[0] == true){ // seno
    strokeWeight(3);
    stroke(colorFunction[0]);
    line(h + r * cos(angle), k, h + r * cos(angle), k - r * sin(angle));
    strokeWeight(1);
    stroke(#B2B2B2);
    line(h, k, h + r * cos(angle), k - r * sin(angle));
  }
  if (states[1] == true){ // coseno
    strokeWeight(3);
    stroke(colorFunction[1]);
    line(h, k, h + r * cos(angle), k);
    strokeWeight(1);
    stroke(#B2B2B2);
    line(h + r * cos(angle), k, h + r * cos(angle), k - r * sin(angle));
  }
  if (states[2] == true){ // tangente
    strokeWeight(3);
    stroke(colorFunction[2]);
    line(h + r, k, h + r, k - r * tan(angle));
    strokeWeight(1);
    stroke(#B2B2B2);
    line(h, k, h + r, k - r * tan(angle));
  }
  if (states[3] == true){ // cotangente
    strokeWeight(3);
    stroke(colorFunction[3]);
    line(h, k - r, h + r * cot(angle), k - r);
    strokeWeight(1);
    stroke(#B2B2B2);
    line(h, k, h + r * cot(angle), k - r);
  }
  if (states[4] == true){ // secante
    strokeWeight(3);
    stroke(colorFunction[4]);
    line(h, k, h + r, k - r * tan(angle));
    strokeWeight(1);
    stroke(#B2B2B2);
    line(h + r, k, h + r, k - r * tan(angle));
  }
  if (states[5] == true){ // cosecante
    strokeWeight(3);
    stroke(colorFunction[5]);
    line(h, k, h + r * cot(angle), k - r);
    strokeWeight(1);
    stroke(#B2B2B2);
    line(h, k - r, h + r * cot(angle), k - r);
  }
}
```

```
float cot(float angle){
  float cotangente = 1 / tan(angle);
  return(cotangente);
}

void monitoring(){
  for (int i = 0; i < 6; i++){
    print(buttonState[i]);
    print('\t');
  }
  println(angle);
}
```

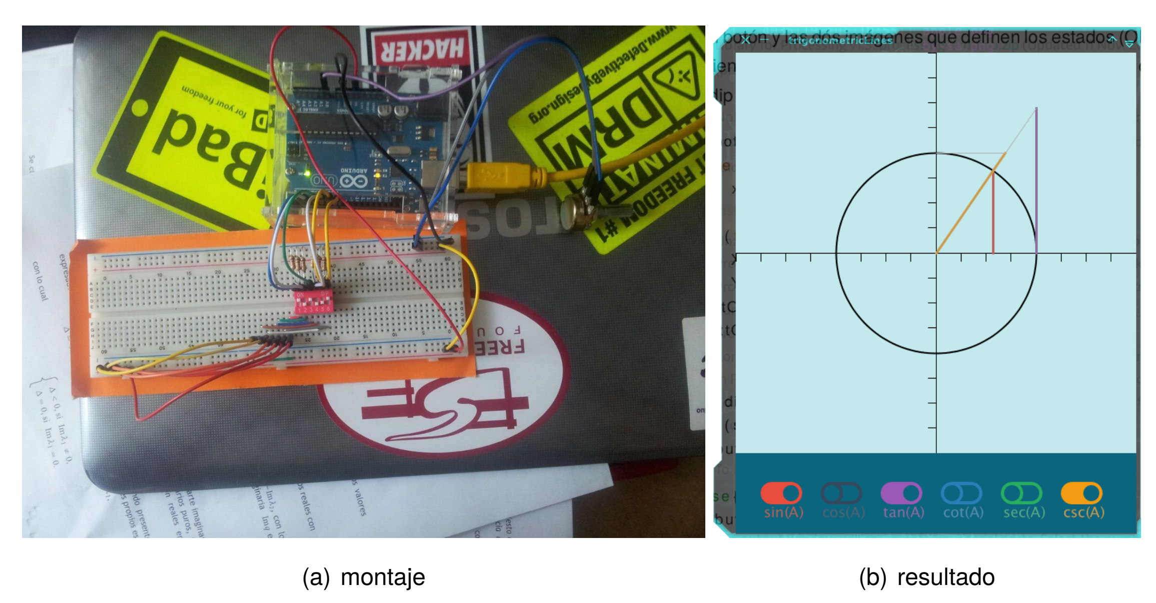
La clase `boton()` del Listing 2, recibe como parámetros de entrada las coordenadas de ubicación del botón y las dós imágenes que definen los estados (ON, OFF) del botón, el método `dibuja()` tiene como parámetro de entrada un valor `boolean` que determina el estado del botón en el dip switch.

```
class boton{
  PImage button, buttOn, buttOff;
  float x, y;

  boton(float X, float Y, PImage on, PImage off){
    x = X;
    y = Y;
    buttOn = on;
    buttOff = off;
  }

  void dibuja(boolean state){
    if (state){
      button = buttOn;
    }
    else{
      button = buttOff;
    }
    image(button, x, y);
  }
}
```

La Figura 1 presenta el montaje y una demostración del resultado.



(a) montaje (b) resultado

Figura 1: Montaje y Resultado del sketch Líneas Trigonométricas

Lic. Fausto Mauricio Lagos Suárez
Mg. Ingeniería computacional y matemática

